

---

# **EECS 349:Machine Learning**

**Bryan Pardo**

## **Topic 2: Decision Trees**

(Includes content provided by: Russel & Norvig, D. Downie, P. Domingos)

# General Learning Task

---

There is a set of possible examples  $X = \{\vec{x}_1, \dots, \vec{x}_n\}$

Each example is an n-tuple of attribute values

$$\vec{x}_1 = \langle a_1, \dots, a_k \rangle$$

There is a target function that maps  $X$  onto some finite set  $Y$

$$f : X \rightarrow Y$$

The DATA is a set of duples <example, target function values>

$$D = \{\langle \vec{x}_1, f(\vec{x}_1) \rangle, \dots, \langle \vec{x}_m, f(\vec{x}_m) \rangle\}$$

Find a **hypothesis**  $h$  such that...

$$\forall \vec{x}, h(\vec{x}) \approx f(\vec{x})$$

# Attribute-based representations

Examples described by **attribute values** (Boolean, discrete, continuous, etc.)  
 E.g., situations where I will/won't wait for a table:

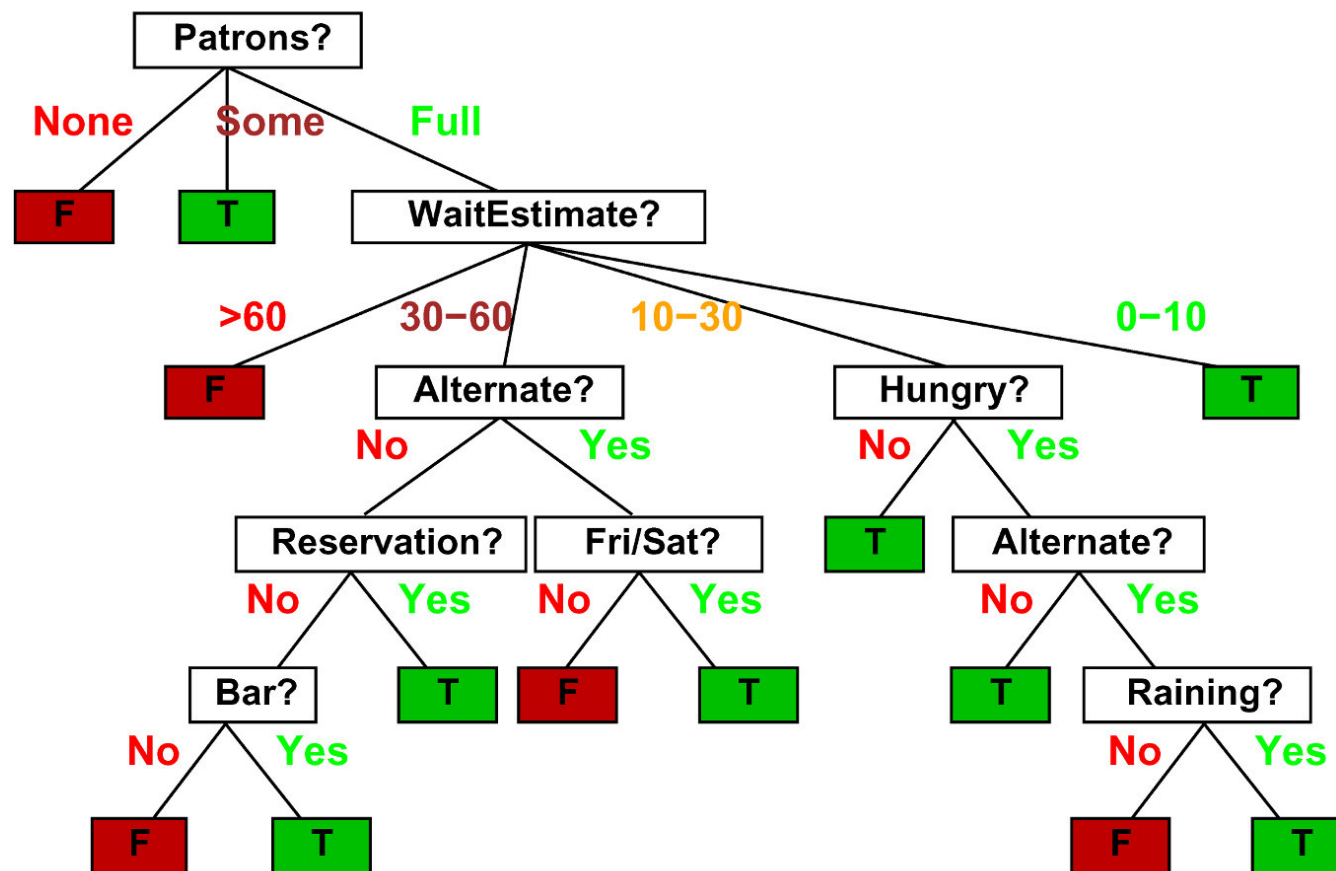
Example	Attributes										Target
	<i>Alt</i>	<i>Bar</i>	<i>Fri</i>	<i>Hun</i>	<i>Pat</i>	<i>Price</i>	<i>Rain</i>	<i>Res</i>	<i>Type</i>	<i>Est</i>	<i>WillWait</i>
$X_1$	<i>T</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>Some</i>	<i>\$\$\$</i>	<i>F</i>	<i>T</i>	<i>French</i>	<i>0-10</i>	<i>T</i>
$X_2$	<i>T</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>Full</i>	<i>\$</i>	<i>F</i>	<i>F</i>	<i>Thai</i>	<i>30-60</i>	<i>F</i>
$X_3$	<i>F</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>Some</i>	<i>\$</i>	<i>F</i>	<i>F</i>	<i>Burger</i>	<i>0-10</i>	<i>T</i>
$X_4$	<i>T</i>	<i>F</i>	<i>T</i>	<i>T</i>	<i>Full</i>	<i>\$</i>	<i>F</i>	<i>F</i>	<i>Thai</i>	<i>10-30</i>	<i>T</i>
$X_5$	<i>T</i>	<i>F</i>	<i>T</i>	<i>F</i>	<i>Full</i>	<i>\$\$\$</i>	<i>F</i>	<i>T</i>	<i>French</i>	<i>&gt;60</i>	<i>F</i>
$X_6$	<i>F</i>	<i>T</i>	<i>F</i>	<i>T</i>	<i>Some</i>	<i>\$\$</i>	<i>T</i>	<i>T</i>	<i>Italian</i>	<i>0-10</i>	<i>T</i>
$X_7$	<i>F</i>	<i>T</i>	<i>F</i>	<i>F</i>	<i>None</i>	<i>\$</i>	<i>T</i>	<i>F</i>	<i>Burger</i>	<i>0-10</i>	<i>F</i>
$X_8$	<i>F</i>	<i>F</i>	<i>F</i>	<i>T</i>	<i>Some</i>	<i>\$\$</i>	<i>T</i>	<i>T</i>	<i>Thai</i>	<i>0-10</i>	<i>T</i>
$X_9$	<i>F</i>	<i>T</i>	<i>T</i>	<i>F</i>	<i>Full</i>	<i>\$</i>	<i>T</i>	<i>F</i>	<i>Burger</i>	<i>&gt;60</i>	<i>F</i>
$X_{10}$	<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>Full</i>	<i>\$\$\$</i>	<i>F</i>	<i>T</i>	<i>Italian</i>	<i>10-30</i>	<i>F</i>
$X_{11}$	<i>F</i>	<i>F</i>	<i>F</i>	<i>F</i>	<i>None</i>	<i>\$</i>	<i>F</i>	<i>F</i>	<i>Thai</i>	<i>0-10</i>	<i>F</i>
$X_{12}$	<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>Full</i>	<i>\$</i>	<i>F</i>	<i>F</i>	<i>Burger</i>	<i>30-60</i>	<i>T</i>

Classification of examples is **positive** (T) or **negative** (F)

# Decision Tree

One possible representation for hypotheses

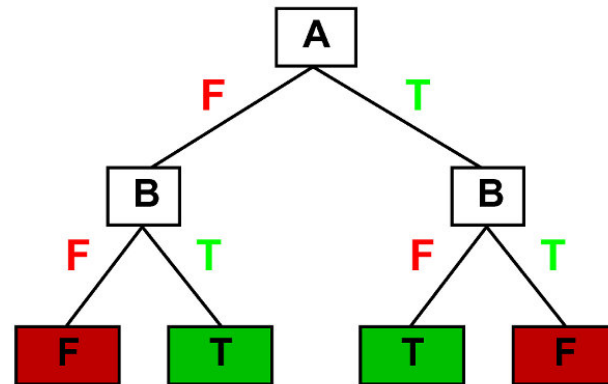
E.g., here is the “true” tree for deciding whether to wait:



# Expressiveness of D-Trees

Decision trees can express any function of the input attributes.  
E.g., for Boolean functions, truth table row  $\rightarrow$  path to leaf:

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F

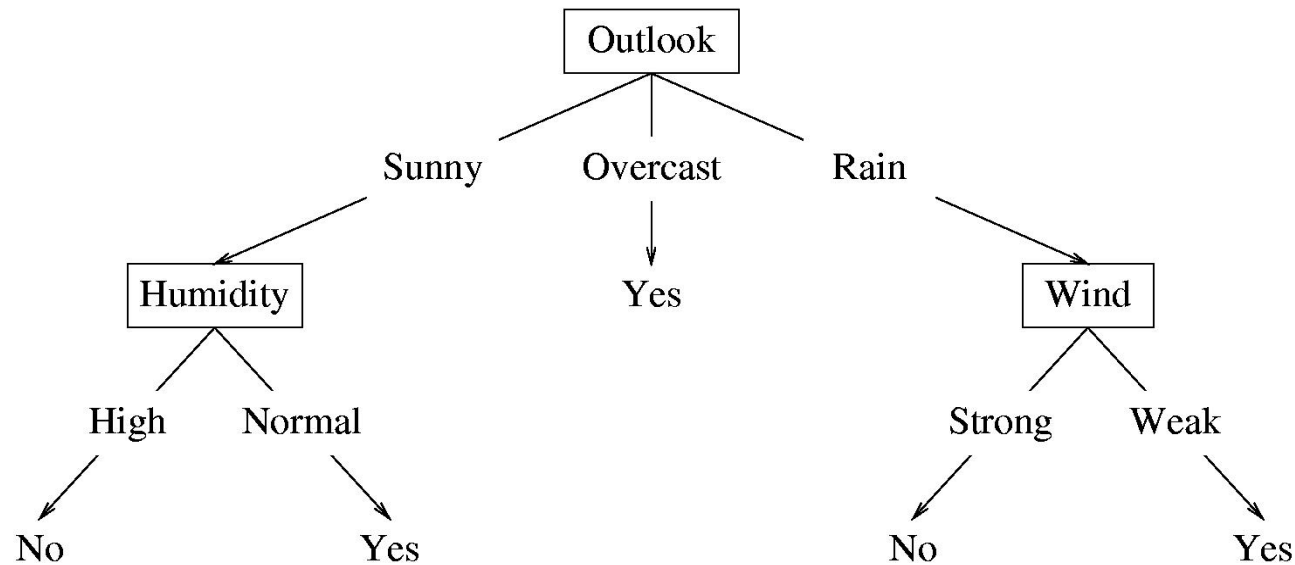


Trivially, there is a consistent decision tree for any training set  
w/ one path to leaf for each example (unless  $f$  nondeterministic in  $x$ )  
but it probably won't generalize to new examples

Prefer to find more **compact** decision trees

# Decision Trees represent *disjunctions of conjunctions*

---



$f(x) = \text{yes}$  iff...

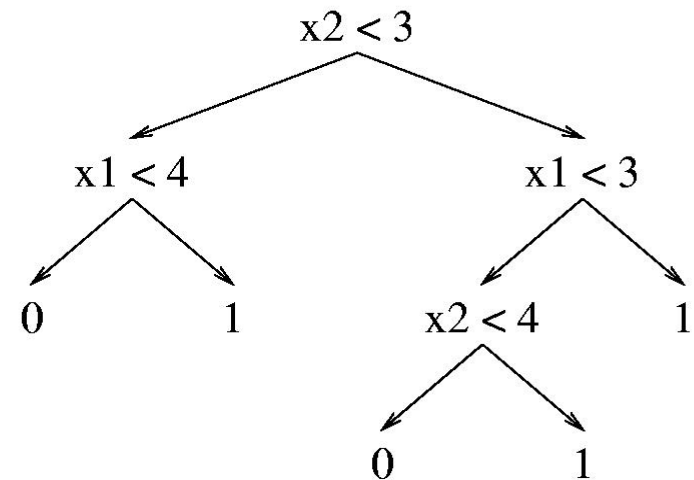
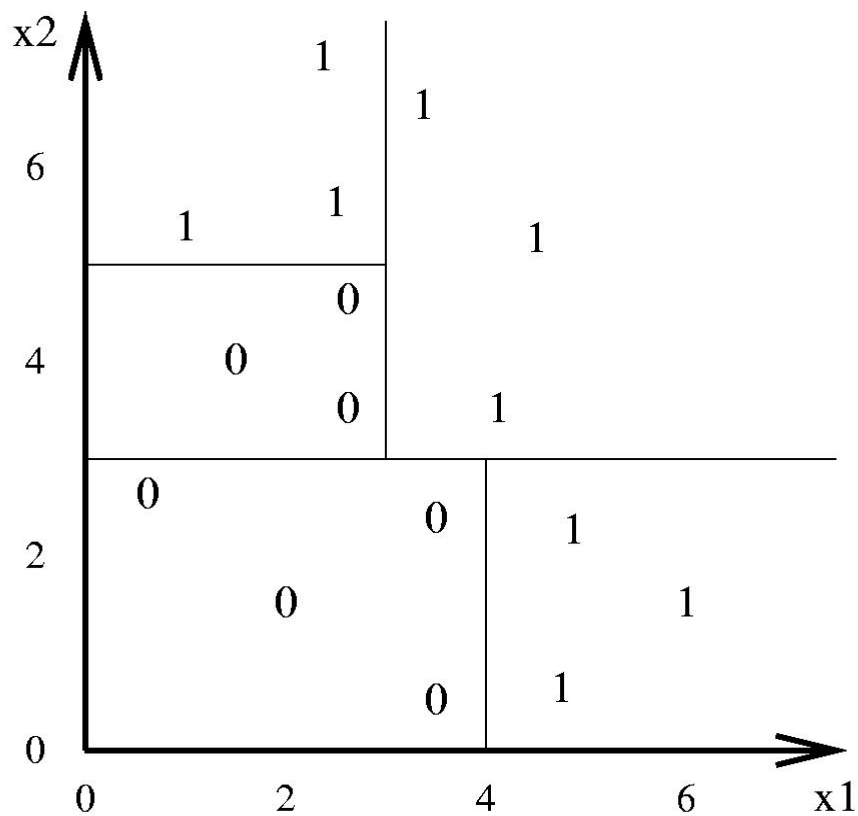
$(\text{Outlook} = \text{Sunny} \wedge \text{Humidity} = \text{Normal}) \vee$

$(\text{Outlook} = \text{overcast}) \vee$

$(\text{Outlook} = \text{rain} \wedge \text{Wind} = \text{weak})$

# Decision Tree Boundaries

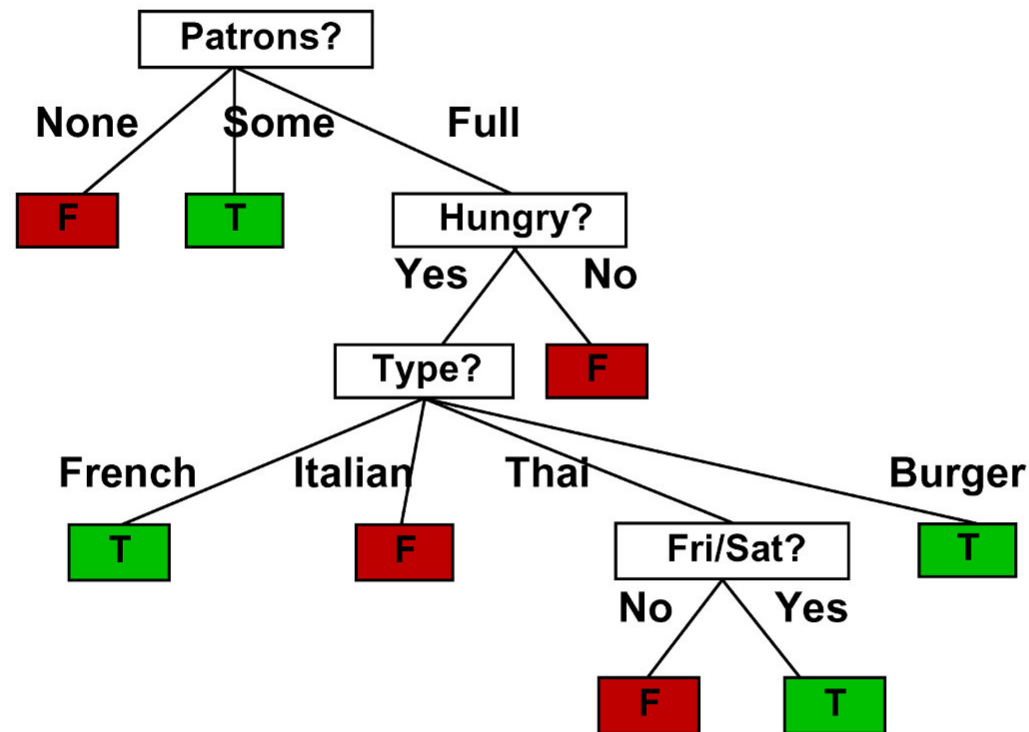
Decision trees divide the feature space into axis-parallel rectangles, and label each rectangle with one of the  $K$  classes.



# A learned decision tree

---

Decision tree learned from the 12 examples:

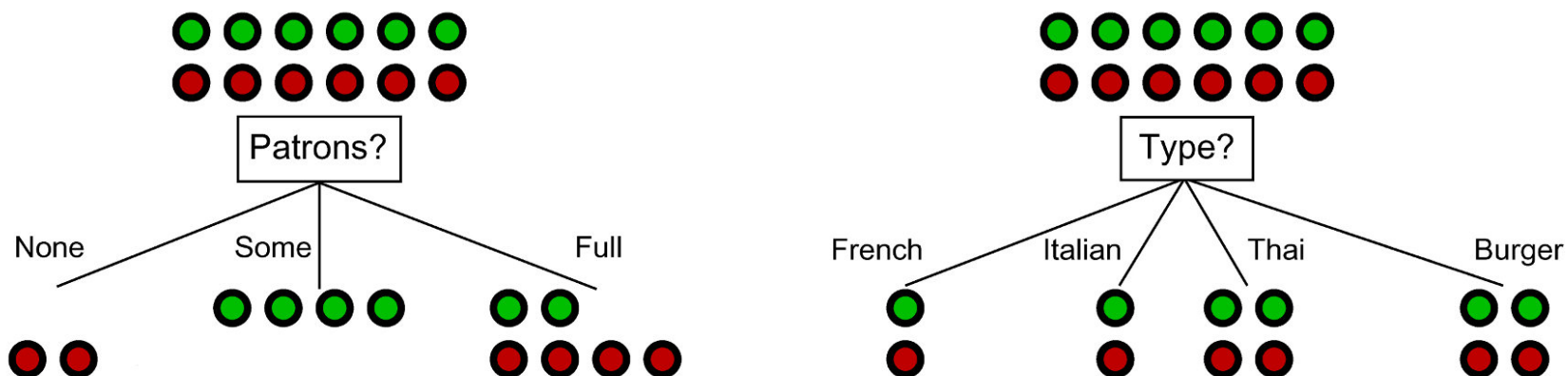


Substantially simpler than “true” tree—a more complex hypothesis isn’t justified by small amount of data



# Choosing an attribute

Idea: a good attribute splits the examples into subsets that are (ideally) “all positive” or “all negative”



*Patrons?* is a better choice—gives **information** about the classification

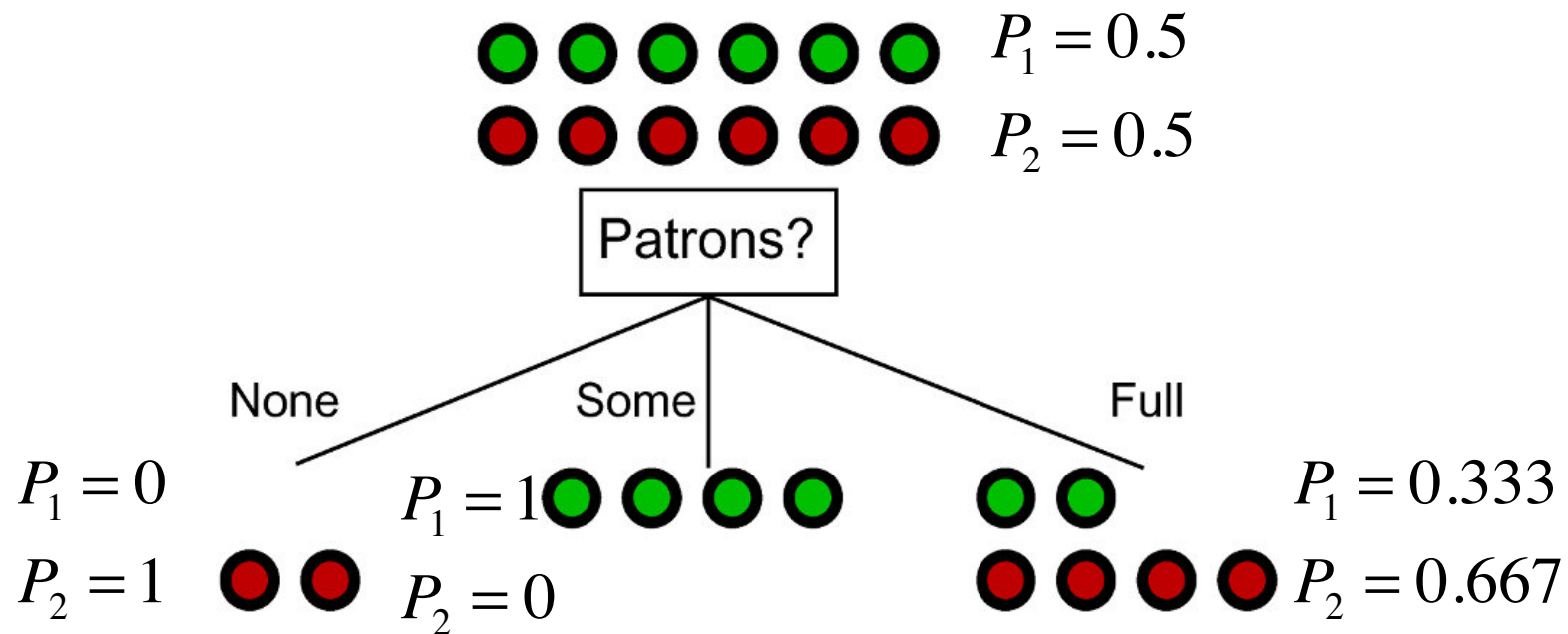
The more skewed the examples in a bin, the better.

We're going to use ENTROPY to as a measure of how skewed each bin is.

# Counts as probabilities

$P_1$  = probability I will wait for a table

$P_2$  = probability I will NOT wait for a table



# Information

---

Information answers questions

The more clueless I am about the answer initially, the more information contained in the answer

Scale: 1 bit = answer to Boolean question with prior  $\langle 0.5, 0.5 \rangle$

Information in an answer when prior is  $\langle P_1, \dots, P_n \rangle$  is

$$H(\langle P_1, \dots, P_n \rangle) = \sum_{i=1}^n -P_i \log_2 P_i$$

(also called **entropy** of the prior)

# About ID3

---

- A recursive, greedy algorithm to build a decision tree
- At each step it picks the best variable to split the data on, and then moves on
- It is “greedy” because it makes the optimal choice at the current step, without considering anything beyond the current step.
- This can lead to trouble, if one needs to consider things beyond a single variable (e.g. multiple variables) when making a choice. (Try it on XOR)

# Decision Tree Learning (ID3)

---

Aim: find a small tree consistent with the training examples

Idea: (recursively) choose “most significant” attribute as root of (sub)tree

```
function DTL(examples, attributes, default) returns a decision tree
  if examples is empty then return default
  else if all examples have the same classification then return the classification
  else if attributes is empty then return MODE(examples)
  else
    best ← CHOOSE-ATTRIBUTE(attributes, examples)
    tree ← a new decision tree with root test best
    for each value  $v_i$  of best do
      examplesi ← {elements of examples with best =  $v_i$ }
      subtree ← DTL(examplesi, attributes − best, MODE(examples))
      add a branch to tree with label  $v_i$  and subtree subtree
    return tree
```

# Choosing an attribute in ID3

---

- For each attribute, find the entropy  $H$  of the example set AFTER splitting on that example
  - \*note, this means taking the entropy of each subset created by splitting on the attribute, and then combining these entropies...weighted by the size of each subset.
- Pick the attribute that creates the lowest overall entropy.

# Entropy prior to splitting

---

Instances where I waited   
Instances where I didn't 

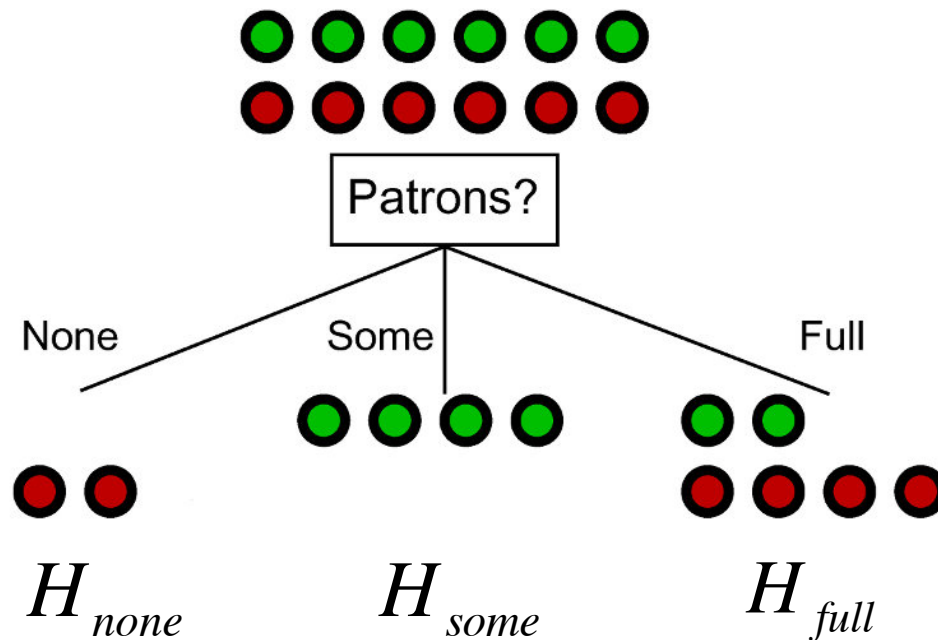
$P_1$  = probability I will wait for a table

$P_2$  = probability I will NOT wait for a table

$$\begin{aligned} H_0 \langle P_1, P_2 \rangle &= \sum_j -P_j \log_2 P_j \\ &= -P_1 \log_2 P_1 - P_2 \log_2 P_2 \\ &= 1 \end{aligned}$$

# If we split on Patrons?

---

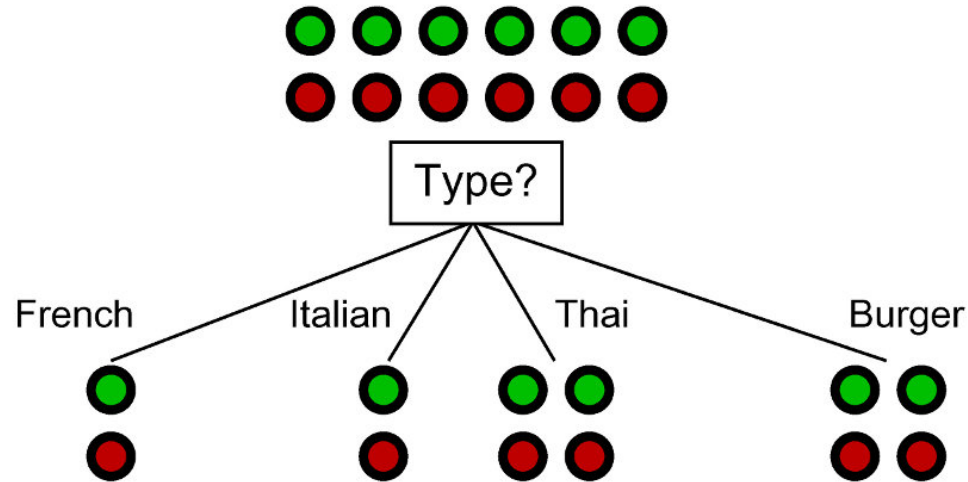


$$\begin{aligned} H_{Patrons} &= W_{none} H_{none} + W_{some} H_{some} + W_{full} H_{full} \\ &= \frac{2}{12} 0 + \frac{4}{12} 0 + \frac{6}{12} \left( -\frac{2}{6} \log_2 \frac{2}{6} - \frac{4}{6} \log_2 \frac{4}{6} \right) = .459 \end{aligned}$$



# If we split on Type

---



$$\begin{aligned} H_{Type} &= W_{french} H_{french} + W_{italian} H_{italian} + W_{thai} H_{thai} + W_{burger} H_{burger} \\ &= \frac{2}{12} 1 + \frac{2}{12} 1 + \frac{4}{12} 1 + \frac{4}{12} 1 = 1 \end{aligned}$$

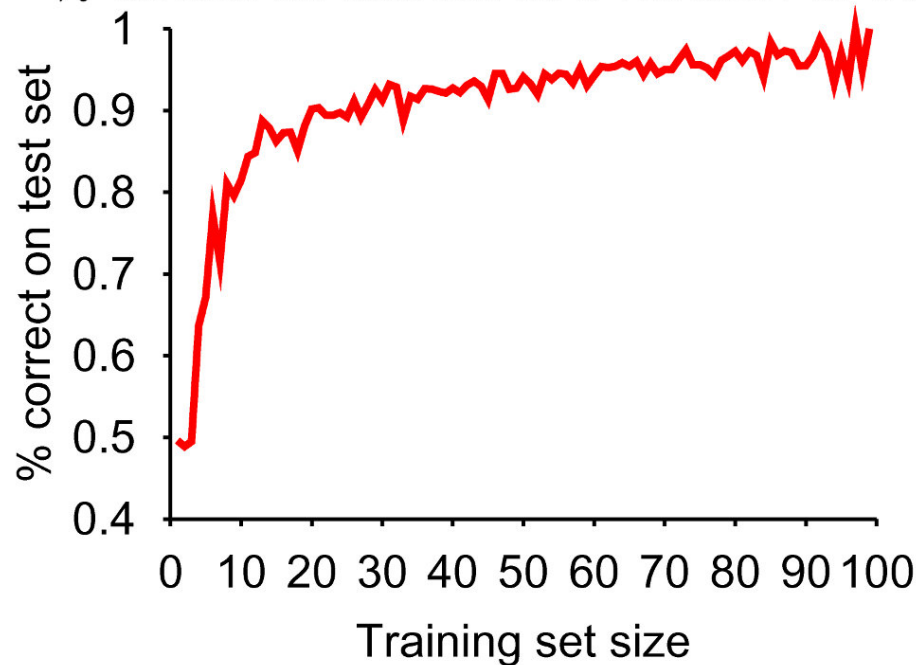
# Measuring Performance

---

How do we know that  $h \approx f$ ? (Hume's **Problem of Induction**)

- 1) Use theorems of computational/statistical learning theory
- 2) Try  $h$  on a new **test set** of examples  
(use **same distribution over example space** as training set)

**Learning curve** = % correct on test set as a function of training set size

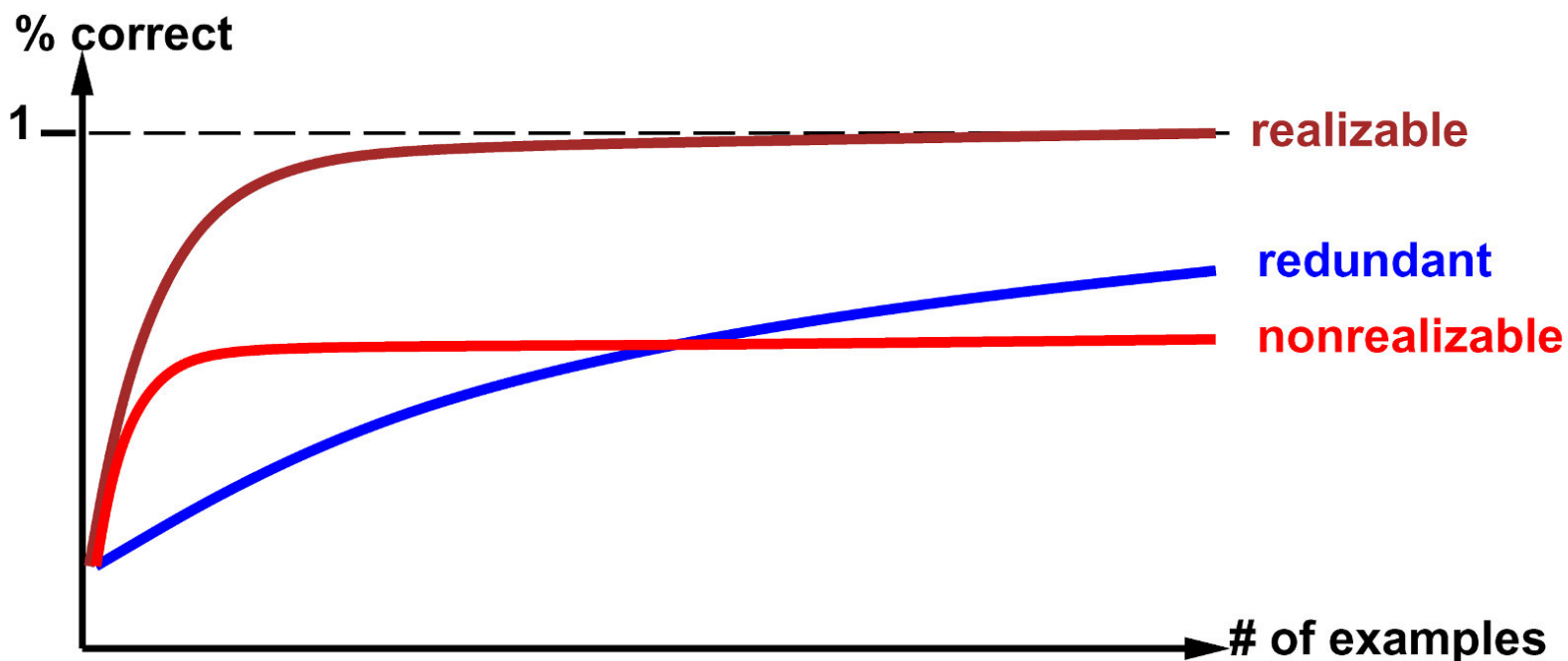


# What the learning curve tells us

---

Learning curve depends on

- **realizable** (can express target function) vs. **non-realizable**  
non-realizability can be due to missing attributes  
or restricted hypothesis class (e.g., thresholded linear function)
- redundant expressiveness (e.g., loads of irrelevant attributes)



## Rule #2 of Machine Learning

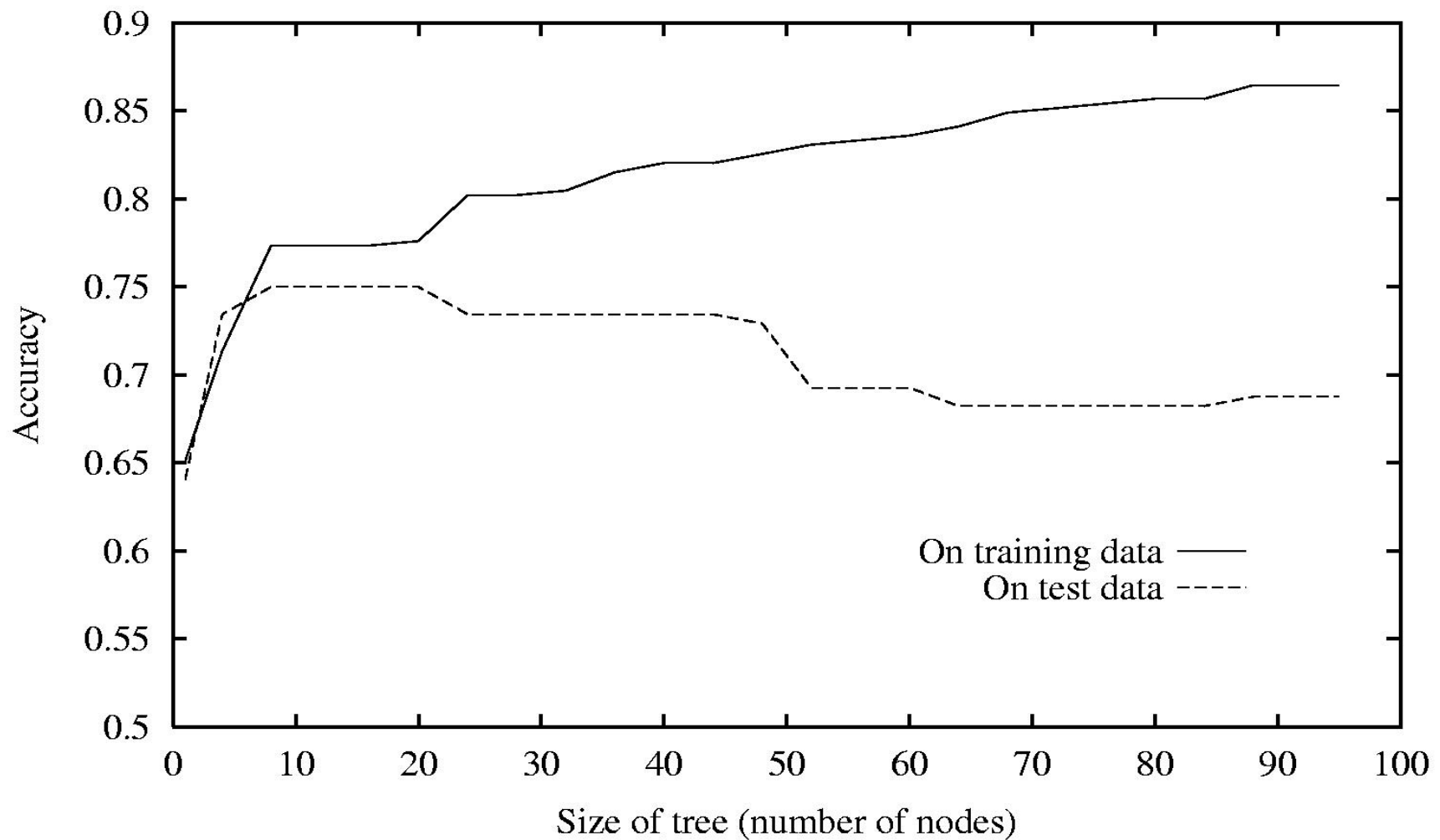
---

The *best* (i.e. the one that generalizes well) hypothesis almost never achieves 100% accuracy on the training data.

(Rule #1 was: you can't learn anything without inductive bias)

# Overfitting

---



# Avoiding Overfitting

---

- Approaches
  - Stop splitting when information gain is low or when split is not statistically significant.
  - Grow full tree and then **prune** it when done
- How to pick the “best” tree?
  - Performance on training data?
  - Performance on validation data?
  - Complexity penalty?

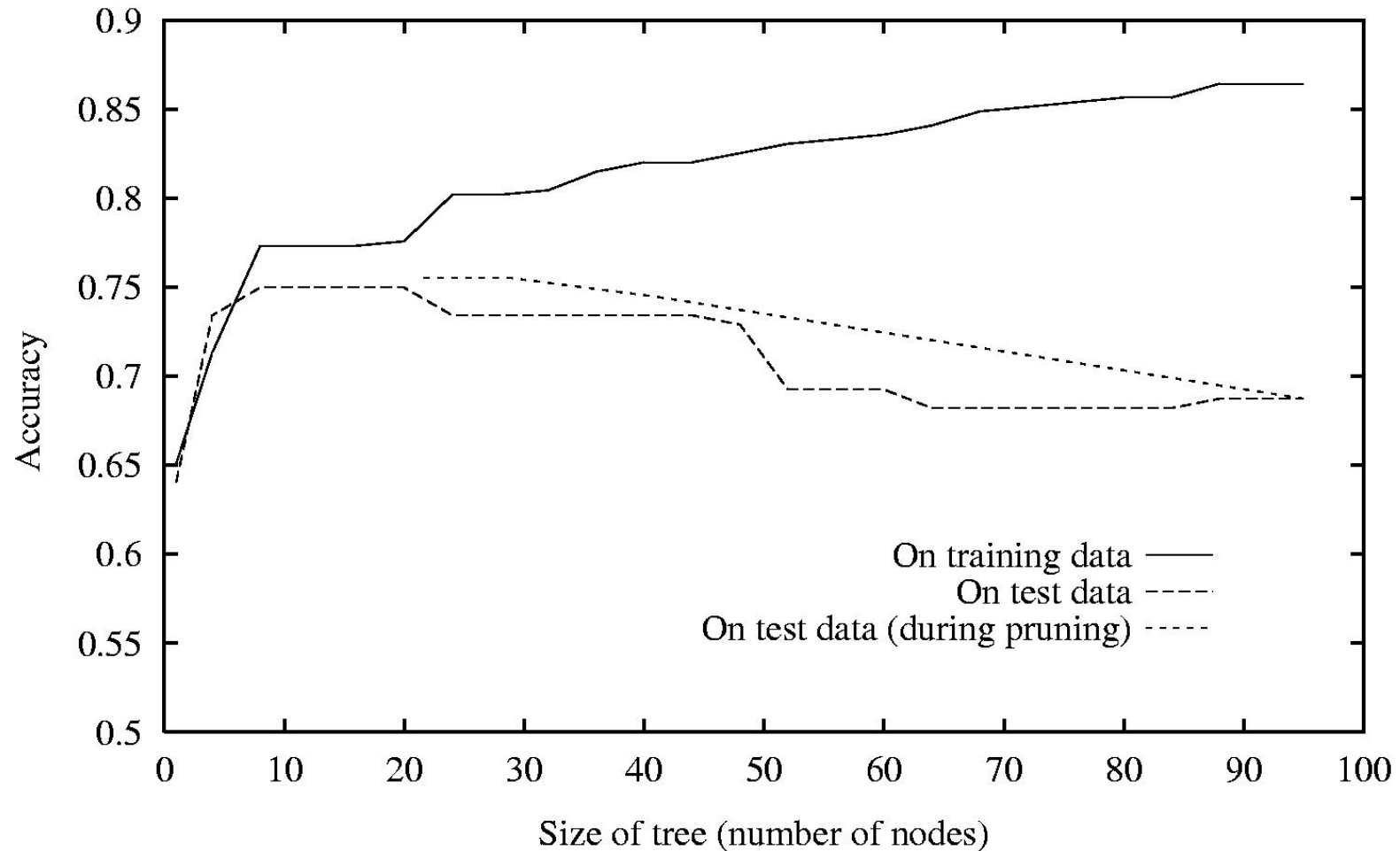
## Reduced-Error Pruning

Split data into *training* and *validation* set

Do until further pruning is harmful:

1. Evaluate impact on *validation* set of pruning each possible node (plus those below it)
2. Greedily remove the one that most improves *validation* set accuracy

# Effect of Reduced Error Pruning





# C4.5 Algorithm

---

- Builds a decision tree from labeled training data
- Also by Ross Quinlan
- Generalizes ID3 by
  - Allowing continuous value attributes
  - Allows missing attributes in examples
  - Prunes tree after building to improve generality

# Rule post pruning

---

- Used in C4.5
- Steps
  1. Build the decision tree
  2. Convert it to a set of logical rules
  3. Prune each rule independently
  4. Sort rules into desired sequence for use

## Scaling Up

- ID3, C4.5, etc. assume data fits in main memory  
(OK for up to hundreds of thousands of examples)
- SPRINT, SLIQ: multiple sequential scans of data  
(OK for up to millions of examples)
- VFDT: at most one sequential scan  
(OK for up to billions of examples)

# Take away about decision trees

---

- Used as classifiers
- Supervised learning algorithms (ID3, C4.5)
- (mostly) Batch processing
- Good for situations where
  - The classification categories are finite
  - The data can be represented as vectors of attributes
  - You want to be able to UNDERSTAND how the classifier makes its choices